

## Contents

<b>1 Routine/Function Prologues</b>	<b>2</b>
1.0.1 maketiles_gds() (Source File: maketiles_gds.F90) . . . . .	2

## 1 Routine/Function Prologues

### 1.0.1 maketiles\_gds() (Source File: maketiles\_gds.F90)

This primary goal of this routine is to determine tile space for GDS based I/O

#### REVISION HISTORY:

```

1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Major F90 and major structure revision
3 Jan 2000: Minor T=0 bug fix, should have no effect on output
8 Mar 2000: Brian Cosgrove; Initialized FGRD to 0 For Dec Alpha Runs
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask
04 Feb 2001: Jon Gottschalck; Added option to read and use Koster tile space

```

#### INTERFACE:

```

subroutine maketiles_gds()
#if ( defined OPENDAP )

```

#### USES:

```

use lisdrv_module, only: lis, grid, glbgindex, tile
use grid_module
use spmdMod
use opendap_module, only: opendap_readcard, opendap_data_prefix

```

#### CONTENTS:

```

allocate(revcnts2(0:npes-1))
allocate(displs2(0:npes-1))

lnr = lis%d%gnr / npes
lnr_last = lis%d%gnr - lnr*(npes-1)

displs2(0) = 0
do t = 1, npes-1
    displs2(t) = displs2(t-1) + lnr*lis%d%gnc
enddo

do t = 0, npes-2
    revcmts2(t) = lnr*lis%d%gnc
enddo
revcmts2(npes-1) = lnr_last*lis%d%gnc

!wlon = 1
wlon = 1 + (lis%d%ic-1) * lis%d%gnc
!elon = lis%d%gnc
elon = lis%d%ic * lis%d%gnc

```

```

!slat = iam*lnr + 1
slat = iam*lnr + 1 + (lis%d%ir-1) * lnr
if ( iam == npes-1 ) then
  !nlat = lis%d%gnr
  nlat = (iam*lnr + lnr_last) + (lis%d%ir-1) * lnr
else
  !nlat = (iam+1)*lnr
  nlat = (iam+1)*lnr + (lis%d%ir-1) * lnr
endif

if ( iam == npes-1 ) then
  lnr = lnr_last
endif

write(cslat, '(i5)' ) slat
write(cnlat, '(i5)' ) nlat
write(cwlon, '(i5)' ) wlon
write(celon, '(i5)' ) elon
write(ciam, '(i3)' ) iam

print*, 'DBG: maketiles_gds -- slat ', slat, ', iam, '
print*, 'DBG: maketiles_gds -- nlat ', nlat, ', iam, '
print*, 'DBG: maketiles_gds -- wlon ', wlon, ', iam, '
print*, 'DBG: maketiles_gds -- elon ', elon, ', iam, '
print*, 'DBG: maketiles_gds -- cslat ', cslat, ', iam, '
print*, 'DBG: maketiles_gds -- cnlat ', cnlat, ', iam, '
print*, 'DBG: maketiles_gds -- cwlon ', cwlon, ', iam, '
print*, 'DBG: maketiles_gds -- celon ', celon, ', iam, '
print*, 'DBG: maketiles_gds -- ciam ', ciam, ', iam, '

call opendap_readcard()
lis%p%myfile = trim(opendap_data_prefix)//'/'// &
               trim(adjustl(ciam))//'/''//lis%p%myfile
lis%p%vfile = trim(opendap_data_prefix)//'/'// &
               trim(adjustl(ciam))//'/''//lis%p%vfile
lis%p%elevfile = trim(opendap_data_prefix)//'/'// &
                  trim(adjustl(ciam))//'/''//lis%p%elevfile

allocate(lat(lis%d%gnc,lnr), stat=ierr)
call check_error(ierr,'Error allocating lat.',iam)

allocate(lon(lis%d%gnc,lnr), stat=ierr)
call check_error(ierr,'Error allocating lon.',iam)

allocate(fgrd(lis%d%gnc,lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating fgrd.',iam)

allocate(mask(lis%d%gnc, lnr), stat=ierr)

```

```

call check_error(ierr,'Error allocating mask.',iam)

allocate(pveg(lis%d%gnc,lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating pveg.',iam)

allocate(tsum(lis%d%gnc, lnr), stat=ierr)
call check_error(ierr,'Error allocating tsum.',iam)

allocate(veg(lis%d%gnc, lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating veg.',iam)

tsum = 0.0

nchp = lis%d%glbnch
print*, 'MSG: maketiles -- Retrieving UMD mask file ', &
trim(lis%p%ofile), '(,iam,)'
call system("opendap_scripts/getmask.pl "//ciam//" //&
trim(lis%p%ofile)//" "// &
cslat//" //cnlat//" //cwlon//" //celon)
print*, 'MSG: maketiles -- Reading ',trim(lis%p%ofile), &
'(,iam,)'
open(30,file=lis%p%ofile,form='unformatted',status='old')
read(30) lat
read(30) lon
read(30) mask
close(30)
call lis_log_msg('MSG: maketiles -- Done reading '//trim(lis%p%ofile))

!-----
! Read in elevation correction file
!-----

if ( lis%f%ecor == 1 ) then
  allocate(elevdiff(lis%d%gnc,lnr), stat=ierr)
  call check_error(ierr,'Error allocating elev diff.',iam)

elevdiff = 0.0

call lis_log_msg('MSG: maketiles -- Retrieving UMD elevation ' //&
'difference file '// trim(lis%p%elevfile))
call system("opendap_scripts/getelev.pl "//ciam//" //&
trim(lis%p%elevfile)//" "// &
cslat//" //cnlat//" //cwlon//" //celon)

open(21,file=lis%p%elevfile,form='unformatted',&
status='old',iostat=ierr)

if (ierr /= 0) then
  call lis_log_msg('ERR: maketiles -- Stop! Problem opening '// &

```

```

            'elevation difference file.')
call lis_log_msg('ERR: maketiles -- Try running without elevation // &
                 'correction option.')
call endrun
else
    read (21) elevdiff
endif
close(21)

call lis_log_msg('MSG: maketiles -- done reading elevation // &
                  'difference file')
endif

!-----
! Select which tile-space veg. info to use (UMD or Koster)
!-----
! if (lis%p%koster .lt. 1) then           ! Use original UMD indexing
print*, 'MSG: maketiles -- Retrieving UMD veg file ', &
trim(lis%p%vfile), ', (',iam,',')
call system("opendap_scripts/getveg.pl "//ciam//" // &
trim(lis%p%vfile)//" "// &
cslat//" //cnlat//" //cylon//" //celon)
print*, 'MSG: maketiles -- Reading ',trim(lis%p%vfile), &
', (',iam,',')
open(98,file=lis%p%vfile,form='unformatted')
read(98) lat
read(98) lon
do t = 1, lis%p%nt
    read(98) veg(:,:,t)
enddo
do r=1,lnr
    do c=1,lis%d%gnc
        isum=0.0
        do t=1,lis%p%nt
            isum=isum+veg(c,r,t)
        enddo
        do t=1,lis%p%nt
            fgrd(c,r,t)=0.0
            if(isum.gt.0) fgrd(c,r,t)=veg(c,r,t)/isum
        enddo
    enddo
enddo
close(98)
deallocate(veg)
call lis_log_msg('MSG: maketiles -- Done reading '//trim(lis%p%vfile))

! endif
!-----
```

```

! Exclude tiles with MINA (minimum tile grid area),
! normalize remaining tiles to 100%
!-----
print*, 'DBG: maketiles -- re-normalizing tiles', ',iam,'
do r=1,lnr
  do c=1,lis%d%gnc
    rsum=0.0
    do t=1,lis%p%nt
      if(fgrd(c,r,t).lt.lis%d%mina)then
        fgrd(c,r,t)=0.0
      endif
      rsum=rsum+fgrd(c,r,t)
    enddo
    if(rsum.gt.0.0) then
      do t=1,lis%p%nt
        if(rsum.gt.0.0)fgrd(c,r,t)=fgrd(c,r,t)/rsum
      enddo

      rsum=0.0
      do t=1,lis%p%nt
        rsum=rsum+fgrd(c,r,t)
      enddo

      if(rsum.lt.0.9999.or.rsum.gt.1.0001)then
        write(*,*) 'error1 in vegetation tiles',rsum,c,r
      endif
    endif
  enddo
enddo
!-----
! Exclude tiles with MAXT (Maximum Tiles per grid),
! normalize remaining tiles to 100%
! Determine the grid predominance order of the tiles
! PVEG(NT) will contain the predominance order of tiles
!-----
print*, 'DBG: maketiles -- excluding MAXT tiles', ',iam,'
do r=1,lnr
  do c=1,lis%d%gnc
    do t=1,lis%p%nt
      fvt(t)=fgrd(c,r,t)
      pveg(c,r,t)=0
    enddo
    do i=1,lis%p%nt
      max=0.0
      t=0
      do j=1,lis%p%nt
        if(fvt(j).gt.max)then
          if(fgrd(c,r,j).gt.0) then

```

```

        max=fvt(j)
        t=j
    endif
    endif
enddo
if(t.gt.0) then
    pveg(c,r,t)=i
    fvt(t)=-999.0
endif
enddo
enddo
!-----
! Impose MAXT Cutoff
!-----
print*,'DBG: maketiles -- imposing MAXT cut-off','(,iam,)'
do r=1,lnr
    do c=1,lis%d%gnc
        rsum=0.0
        do t=1,lis%p%nt
            if(pveg(c,r,t).lt.1) then
                fgrd(c,r,t)=0.0
                pveg(c,r,t)=0
            endif
            if(pveg(c,r,t).gt.lis%d%maxt) then
                fgrd(c,r,t)=0.0          ! impose maxt cutoff
                pveg(c,r,t)=0
            endif
            rsum=rsum+fgrd(c,r,t)
        enddo
    enddo
!-----
! Renormalize veg fractions within a grid to 1
!-----
if(rsum.gt.0.0) then
    do t=1,lis%p%nt
        if(rsum.gt.0.0)fgrd(c,r,t)= fgrd(c,r,t)/rsum
    enddo

    rsum=0.0
    do t=1,lis%p%nt
        rsum=rsum+ fgrd(c,r,t)  !recalculate rsum to check
    enddo
    tsum(c,r)=rsum

    if(rsum.lt.0.9999.or.rsum.gt.1.0001)then !check renormalization
        write(*,*) 'error2 in vegetation tiles',rsum,c,r
    endif

```

```

        endif

    enddo
enddo
deallocate(pveg)

call lis_log_msg('DBG: maketiles -- Gathering data onto masterproc')

if ( masterproc ) then
    allocate(glbmask(lis%d%gnc, lis%d%gnr), stat=ierr)
else
    allocate(glbmask(1, 1), stat=ierr)
endif
call check_error(ierr,'Error allocating glbmask.',iam)
if ( npes > 1 ) then
    call lis_log_msg('DBG: maketiles -- Gathering mask')
    call MPI_GATHERV(mask,lis%d%gnc*lnr,MPI_REAL, &
                      glbmask,recvcnts2,displs2,MPI_REAL, &
                      0,MPI_COMM_WORLD, ierr)
else
    glbmask = mask
endif
deallocate(mask)

if ( lis%f%ecor == 1 ) then
    if ( masterproc ) then
        allocate(glbelevdiff(lis%d%gnc, lis%d%gnr), stat=ierr)
    else
        allocate(glbelevdiff(1, 1), stat=ierr)
    endif
    call check_error(ierr,'Error allocating glbelevdiff.',iam)
    if ( npes > 1 ) then
        call lis_log_msg('DBG: maketiles -- Gathering elevdiff')
        call MPI_GATHERV(elevdiff,lis%d%gnc*lnr,MPI_REAL, &
                         glbelevdiff,recvcnts2,displs2,MPI_REAL, &
                         0,MPI_COMM_WORLD, ierr)
    else
        glbelevdiff = elevdiff
    endif
    deallocate(elevdiff)
endif

if ( masterproc ) then
    allocate(glblat(lis%d%gnc, lis%d%gnr), stat=ierr)
else
    allocate(glblat(1, 1), stat=ierr)
endif
call check_error(ierr,'Error allocating glblat.',iam)
```

```

if ( npes > 1 ) then
  call lis_log_msg('DBG: maketiles -- Gathering lat')
  call MPI_GATHERV(lat,lis%d%gnc*lnr,MPI_REAL, &
    glblat,recvcnts2,displs2,MPI_REAL, &
    0,MPI_COMM_WORLD, ierr)
else
  glblat = lat
endif
deallocate(lat)

if ( masterproc ) then
  allocate(glblon(lis%d%gnc, lis%d%gnr), stat=ierr)
else
  allocate(glblon(1, 1), stat=ierr)
endif
call check_error(ierr,'Error allocating glblon.',iam)
if ( npes > 1 ) then
  call lis_log_msg('DBG: maketiles -- Gathering lon')
  call MPI_GATHERV(lon,lis%d%gnc*lnr,MPI_REAL, &
    glblon,recvcnts2,displs2,MPI_REAL, &
    0,MPI_COMM_WORLD, ierr)
else
  glblon = lon
endif
deallocate(lon)

if ( masterproc ) then
  allocate(glbfgrd(lis%d%gnc, lis%d%gnr, lis%p%nt), stat=ierr)
else
  allocate(glbfgrd(1, 1, 13), stat=ierr)
endif
call check_error(ierr,'Error allocating glbfgrd.',iam)
if ( npes > 1 ) then
  call lis_log_msg('DBG: maketiles -- Gathering fgrd')
  do t = 1, lis%p%nt
    call MPI_GATHERV(fgrd(:,:,t),lis%d%gnc*lnr,MPI_REAL, &
      glbfgrd(:,:,t),recvcnts2,displs2,MPI_REAL, &
      0,MPI_COMM_WORLD, ierr)
  enddo
else
  glbfgrd = fgrd
endif
deallocate(fgrd)

if ( masterproc ) then
  allocate(glbtsum(lis%d%gnc, lis%d%gnr), stat=ierr)
else
  allocate(glbtsum(1, 1), stat=ierr)

```

```

        endif
        call check_error(ierr,'Error allocating glbtsum.',iam)
        if ( npes > 1 ) then
            call lis_log_msg('DBG: maketiles -- Gathering tsum')
            call MPI_GATHERV(tsum,lis%d%gnc*lnr,MPI_REAL, &
                glbtsum,recvcnts2,displs2,MPI_REAL, &
                0,MPI_COMM_WORLD, ierr)
        else
            glbtsum = tsum
        endif
        deallocate(tsum)
        deallocate(recvcnts2)
        deallocate(displs2)

        call lis_log_msg('MSG: maketiles -- Beginning masterproc computations')

        if ( masterproc ) then
            landnveg = 5
!           if (lis%p%koster .eq. 1) landnveg = 4
            lis%d%glbnch=0
            do t=1,lis%p%nt
                do r=1,lis%d%gnr
                    do c=1,lis%d%gnc
                        if(glbmask(c,r).gt.0.99.and. &
                            glbmask(c,r).lt.3.01)then
                            if(glbfrd(c,r,t).gt.0.0)then
                                lis%d%glbnch=lis%d%glbnch+1
                            endif
                            if(glbsum(c,r).eq.0.0.and.t.eq.landnveg)then
                                lis%d%glbnch=lis%d%glbnch+1
                            endif
                        endif
                    enddo
                enddo
            enddo

            print*, 'DBG: maketiles -- glbnch',lis%d%glbnch,' (',iam,')'
            allocate(tile(lis%d%glbnch))

            lis%d%glbngrid=0
            do r=1,lis%d%gnr
                do c=1,lis%d%gnc
                    if(glbmask(c,r).gt.0.99 .and. &
                        glbmask(c,r).lt.3.01) then
                        lis%d%glbngrid=lis%d%glbngrid+1
                    endif
                enddo
            enddo
        endif
    enddo
enddo

```

```

count = 1
print*, 'DBG: maketiles1 -- glbnch',lis%d%glbnch,' (',iam,',')
allocate(grid(lis%d%glbngrid))
allocate(glbindex(lis%d%gnc, lis%d%gnr))
print*, 'DBG: maketiles2 -- glbnch',lis%d%glbnch,' (',iam,',')
do r=1,lis%d%gnr
  do c=1,lis%d%gnc
    glbindex(c,r) = -1
    if(glbmask(c,r).gt.0.99 .and. &
        glbmask(c,r).lt.3.01) then
      grid(count)%lat = glblat(c,r)
      grid(count)%lon = glblon(c,r)
      grid(count)%fgrd = glbfgrd(c,r,:)
      glbindex(c,r) = count
      count = count+1
    endif
  enddo
enddo
deallocate(glblat)
deallocate(glblon)
print*, 'DBG: maketiles3 -- glbnch',lis%d%glbnch,' (',iam,',')
count = 0
do r=1,lis%d%gnr
  do c=1,lis%d%gnc
    do t=1,lis%p%nt
      if(glbmask(c,r).gt.0.99.and. &
          glbmask(c,r).lt.3.01)then
        if(glbfrd(c,r,t).gt.0.0)then
          count = count+1
          tile(count)%row=r
          tile(count)%col=c
          tile(count)%index = glbindex(c,r)
          tile(count)%vegt=t
          tile(count)%fgrd=glbfgrd(c,r,t)
          if ( lis%f%ecor == 1 ) then
            if ( glbelevdiff(c,r) == -9999.0 ) then
              glbelevdiff(c,r) = 0.0
            endif
            tile(count)%elev=glbelevdiff(c,r)
          endif
        endif
      endif
      if(gbtsum(c,r).eq.0.0.and.t.eq.landnveg)then
        count=count+1
        tile(count)%row=r
        tile(count)%col=c
        tile(count)%index = glbindex(c,r)
        tile(count)%vegt=t
        tile(count)%fgrd=1.0
      endif
    enddo
  enddo
enddo

```

```

        if ( lis%f%ecor == 1 ) then
            if ( glbelevdiff(c,r) == -9999.0 ) then
                glbelevdiff(c,r) = 0.0
            endif
            tile(count)%elev=glbelevdiff(c,r)
        endif
    endif
    enddo
enddo
print*, 'DBG: maketiles4 -- glbnch',lis%d%glbnch, (',iam,')
deallocate(glbmask)
deallocate(glbfgrd)
deallocate(glbtsum)
if ( lis%f%ecor == 1 ) then
    deallocate(glbelevdiff)
endif
write(*,*) 'MSG: maketiles -- Size of Tile Dimension:',NCHP, &
    (',iam,')
write(*,*) 'MSG: maketiles -- Actual Number of Tiles:', &
    LIS%D%GLBNCH,' (',iam,')
write(*,*)

write(*,*) 'MSG: maketiles -- Size of Grid Dimension:', &
    lis%d%glbngrid,' (',iam,')
write(*,*)

endif
print*, 'MSG: maketiles -- done',' (',iam,')

#endif
return

```